Informatik

Teil 2

Prof. Dr. Aris Christidis

WS 2003 / 04

Zur Organisation

Kontakt@ MNI:

Bodenkunde-Gebäude, Raum I 208
Sprechzeiten: nach Vereinbarung (vorerst)
Tel. 0641-309-2391, -2300 (Sekr. MNI)
A.Christidis@mni.fh-giessen.de

Folien unter:

http://homepages.fh-giessen.de/~hg11237/ – bzw.:
http://www.fh-giessen.de > Fachbereiche
> 06 Mathematik, Naturwissenschaften und Informatik
> Professoren > Christidis, Aristovoulos Dr. > www

Übungen:

- Alle Übungen (EW: Teile 1&2) zu 100% gelöst ⇨
  10% der Punkte für die Klausur-Note „Sehr gut“ (1,0).
- Keine (positiven) Benachrichtigungen vorgesehen!

Zum Fach - Literatur:

- W. Doberenz, Th. Kowalski:
  “Programmieren lernen in Visual Basic 6”
  Hanser 1999, 336 S.,
  ISBN 3-446-19596-3, € 24,90

- Regionales Rechenzentrum für Niedersachsen (RRZN) / Softwareberatung Herdt:
  “Visual Basic 6.0 - Grundlagen”
  RRZN Hannover 1999, ca. € 5,50 (DM 10,80)
  (http://www.rrzn.uni-hannover.de/Dokumentation/Handbuecher/index.html)

- Regionales Rechenzentrum für Niedersachsen (RRZN) / Softwareberatung Herdt:
  “Grundlagen der Programmierung”
  RRZN Hannover 1999, ca. € 5,30 (DM 10,30)
  (http://www.rrzn.uni-hannover.de/Dokumentation/Handbuecher/index.html)

Zum Fach - Literatur:

- Walter Doberenz, Thomas Kowalski:
  „**Visual Basic 6 - Grundlagen und Profiwissen“**
  Hanser 1999, 1072 S.,
  ISBN 3-446-19594-7, € 49,90

- Regionales Rechenzentrum für Niedersachsen (RRZN) / Softwareberatung Herdt:
  “Visual Basic 6.0 – Fortgeschrittene Programmierung”
  RRZN Hannover 1999, ca. € 5,50
  (http://www.rrzn.uni-hannover.de/Dokumentation/Handbuecher/index.html)

- David I. Schneider:
  „**An Introduction to Programming with Visual Basic 6.0“**
  (Including Microsoft Visual Basic 6.0 Working Model)
  Prentice Hall 1999, ca.815 S.,
  ISBN 0139364285

Zum Fach - Literatur:

- Sehr interessante Tutorien:
  www.vbinformation.com/tutor.htm

- reichhaltig (Code u. Tutorials):
  www.freevbcode.com

- Viel Code: Visual Basic Developers Resource Centre bei Microsoft Most Valuable Professionals (MVPs)
  www.mvps.org

- Forum: Visual Basic Web Magazine (VBWM)
  www.vbwm.com/learnvb/tutorials.asp

Vom Algorithmus zum Programm

Auf der Grundlage eines Programms kann ein Rechner

- **Daten** mit seiner Umgebung **austauschen**,
- mathematische und logische **Operationen durchführen**,
- in Abhängigkeit von anfallenden Daten den **Programmablauf** zur Laufzeit **verändern**.

Er kann so mathematisch-logische Aufgaben lösen, wenn deren Lösungsweg entsprechend formuliert werden kann.

Derartige formale Beschreibungen von Arbeitsschritten werden **Algorithmen** genannt.

Vom Algorithmus zum Programm

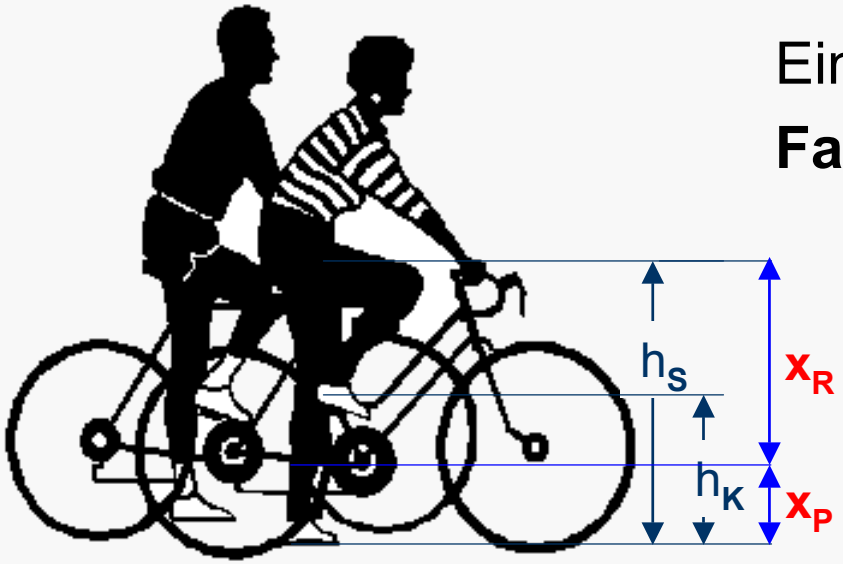
Algorithmus-Definition:

Ein (deterministischer) Algorithmus ist eine **formale Beschreibung** einer Verarbeitungsvorschrift zur Lösung einer bestimmten Art von Problemen, die folgende Bedingunen erfüllt: *

- **Exaktheit** (präzise, eindeutig),
- **Finitheit** (endliche Länge),
- **Vollständigkeit** (für alle Einzelfälle),
- **Effektivität** (elementare, real ausführbare Einzelschritte),
- **Terminierung** (Resultat nach endlich vielen Schritten),
- **Determinismus** (Reproduzierbarkeit).

Vom Algorithmus zum Programm

Prof. Dr. Aris Christidis



Einfaches Berechnungsprogramm:

Fahrrad-Design nach Körpermaß

Eingaben: Sitzhöhe h_S, Kniehöhe h_K

⇨ Ausgaben: Höhe Rahmen x_R, Länge Pedalkurbel x_P

(Gl. A) $x_R + x_P = h_S$

(Gl. B) $x_R - x_P = h_K$

(A) + (B) ⇨ $2 \cdot x_R = h_S + h_K$

(A) – (B) ⇨ $2 \cdot x_P = h_S - h_K$

Variable, Zuweisung, Konstante, math.Operationen

⇨ $x_R = h_S / 2 + h_K / 2$

⇨ $x_P = h_S / 2 - h_K / 2$

bzw.

$x_R = (h_S + h_K) / 2$

$x_P = (h_S - h_K) / 2$

mathematisch äquivalent

Vom Algorithmus zum Programm

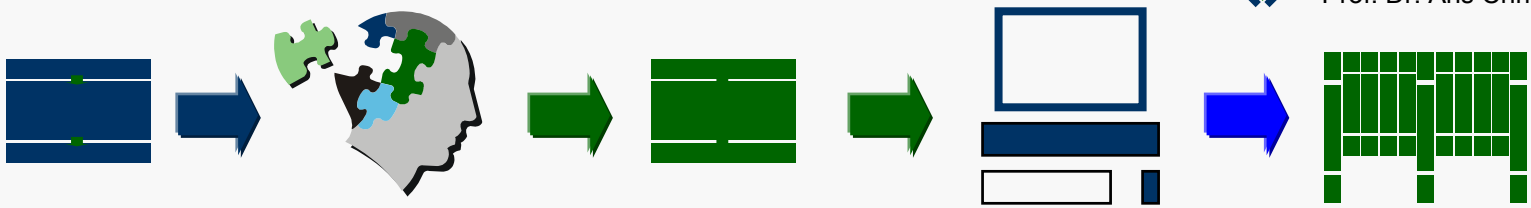
Prof. Dr. Aris Christidis

Ansatz 1:	Ansatz 2:	Ansatz 3:
Speicher: h_S, h_K, x_R, x_P	Speicher: h_S, h_K, x_R, x_P	Speicher: h_S,h_K,x_R,x_P,v_S,v_K
1.Lies Zahlen h_S, h_K ein	1.Lies Zahlen h_S, h_K ein	1. Lies Zahlen h_S, h_K ein
2.Berechne die Zahl $x_R = h_S / 2 + h_K / 2$	2.Berechne die Zahl $x_R = (h_S + h_K) / 2$	2. Berechne $v_S = h_S / 2$
		3. Berechne $v_K = h_K / 2$
3.Berechne die Zahl $x_P = h_S / 2 – h_K / 2$	3.Berechne die Zahl $x_P = (h_S – h_K) / 2$	4. Berechne $x_R = v_S + v_K$
		5. Berechne $x_P = v_S – v_K$
4.Gib x_R und x_P aus	4.Gib x_R und x_P aus	6. Gib x_R und x_P aus
● Zuweisungn: 2 — *t = 2*	● Zuweisungn: 2 — *t = 2*	● Zuweisungen: 4 — *t = 4*
● Add./Subtr.: 2 — *26*	● Add./Subtr.: 2 — *26*	● Add./Subtr.: 2 — *26*
● Divisionen: 4 — *84*	● Divisionen: 2 — *42*	● Divisionen: 2 — *42*
Σ_t:112	***Σ_t : 70***	***Σ_t : 72***
● max.Wert: h_S	● max.Wert: $h_S + h_K$	● max.Wert: h_S
● Variablen: 4	● Variablen: 4	● Variablen: 6

Takte (z.B.)

Vom Algorithmus zum Programm

Prof. Dr. Aris Christidis



Informeller Lösungsweg

Formaler Algorithmus

Programmcode

Maschinencode

Die Sprachfamilien:

- **Maschinensprachen**
  (seit Mitte der 40er)
- **Assembler**
  (maschinenorientiert, seit Anfang der 50er)
- **Hochsprachen**
  (problemorientiert, ab Mitte der 50er)

Code-Beispiel: 3 + 4 = ?

```
0001 1010 0011 0100
```

```
ADD 3, 4
```

```
Summe = 3 + 4;
```

Visual Basic als Programmiersprache

- Visual Basic
  - Visual Zum Erstellen graphischer Benutzeroberflächen
  - Basic Beginners All-Purpose Symbolic Instruction Code
- Visual Basic ist eine Programmiersprache ...
  - ... für Programme mit Schwerpunkt auf graphischen Ein-/ Ausgaben
  - ... für Microsoft-Office-Anwendungen wie Excel, Word, Access, ...
  - ... mit objektorientierten Ansätzen
  - ... die leicht zu erlernen ist und (daher) Spaß macht
- Visual Basic ist nicht ...
  - ... die 'geniale' Universalsprache
  - ... schnell

Visual Basic als Programmiersprache

Vorteile:

- VB kapselt die Windows-Schnittstellen (Windows-Application Programming Interfaces • API)
- VB besitzt eine interaktive IDE und Laufzeitumgebung
- VB ist in alle Office-Produkte eingebaut u. erweiterbar

Nachteile:

- VB ist langsamer als C, auch wenn sie compiliert ist,
- VB ist nur mit Laufzeitbibliothek lauffähig (Lizenzen, Größe !)
- VB ist nicht systemnah

Visual Basic als Programmiersprache

VB ist gut für

- Datenbank-Front-Ends für SQL-Server
- Eigenständige Datenbank-Applikationen
- Inhouse-Lösungen
- Lernprogramme, einfache Spiele
- Office-Applikationen

VB ist nicht so gut für

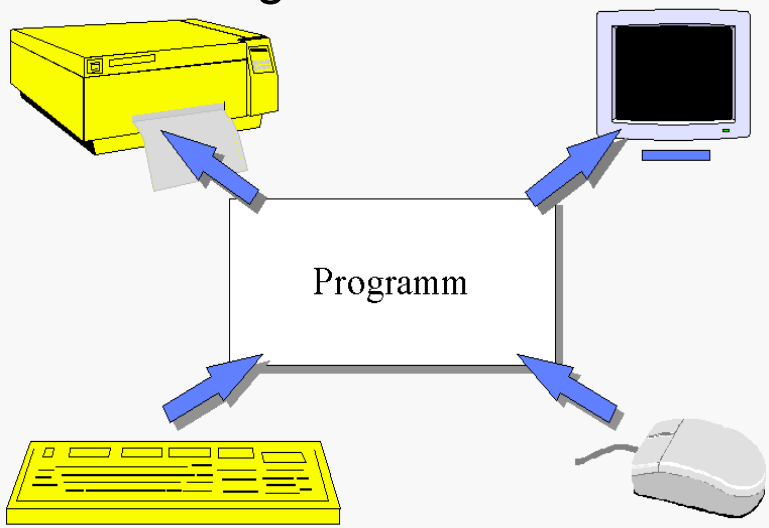
- Extrem zeitkritische Anwendungen
- Hardware-nahe Programmierung
- Plattformunabhängige Anwendungen
- Massenanwendungen (wg. Laufzeitbibliothek)

Betriebssysteme

Prof. Dr. Aris Christidis

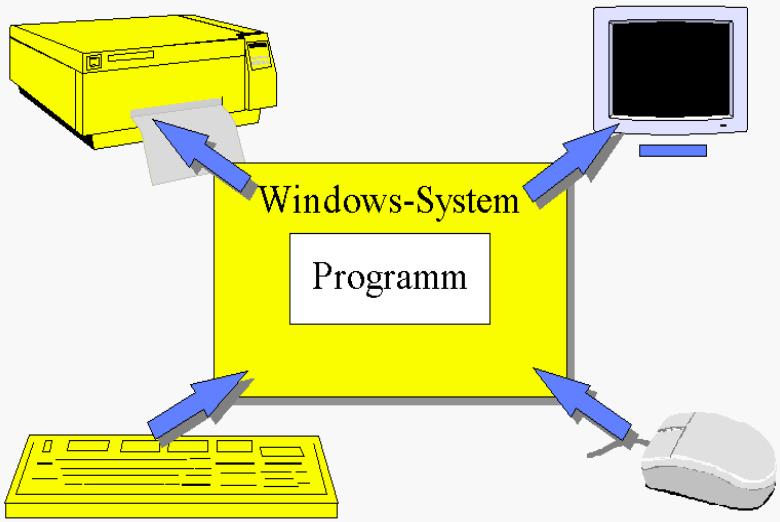
Anwendungsprogrammierung unter **MS-DOS**:

"Standard-I/O" u.standardisierte Rechenzeit-Zuweisung; abweichende Handhabung durch Programmierer/in



Windows-Programmierung:

Kommunikation mit Peripherie ins System integriert: Geräte werden wie "unter Windows" angesprochen.



Bilder: W. Doberenz, Th. Kowalski: "Programmieren lernen in Visual Basic 5", Hanser 1997

Zur Verdeutlichung: Word for DOS auf ≥20 Disketten ausgeliefert; darunter: 1-2 für das Programm selbst, Rest für diverse Treiber

Grundbegriffe der Programmierung

Neuerungen d. Windows-Programmierung (gegenüber DOS):

- **Preemptive Multitasking** („Verdrängender Mehrprozeß-Betrieb“) Teilung der Rechenzeit durch mehrere Anwendungsprogramme unter der Kontrolle des Betriebssystems

- **Multithreading** („Mehrfach-Handlungsfaden-Verarbeitung“) Gezielte Aufteilung der Rechenzeit zwischen „leichtgewichtigen Prozessen“ einer Applikation

- **Event-driven processing** („Ereignisgesteuerte Arbeitsweise“) Programm-Ablauf abhängig von Ereignissen (=speziellen Zustandsänderungen) wie Tastendruck, Mausklick, Beendigung eines(Teil-)Prozesses etc.

Grundbegriffe der Programmierung

Objekte sind die Elemente der Bedienoberfläche, d.h.:

- **Formulare (Formen):** Windows-Fenster, in denen eine VB-Applikation läuft. Sie können Text, Grafik, Komponenten oder weitere Formulare enthalten.

- **Komponenten** heißen die Elemente der Benutzerschnittstelle, über welche die Kommunikation zwischen Programm und Benutzer erfolgt: Schaltflächen, Bildlaufleisten, Textfelder etc.

- **Sonstige Objekte**, zu denen der Zugriff über Windows läuft – z.B. Bildschirme, Drucker, Zwischenablage u.v.m..

Grundbegriffe der Programmierung

Objekten zugeordnet sind:

- **Eigenschaften:** statische Festlegungen der Objekt-Zustände (Größe, Farbe, Aktivierung etc.)
- **Methoden:** mit der Objekt-Definition abgelegte Funktionen und Prozeduren, die das Objekt-Verhalten bestimmen (nach Maus-Klick, Tasten-Druck etc.)
- **Ereignisse:** Nachrichten, die von Objekten ausgelöst werden und bei anderen Objekten Aktionen hervorrufen

Grundlegendes zum Programmieren mit VB

Etappen der VB-Programmentwicklung:

1. **Bedienoberfläche visuell entwerfen**

   (Vom Editor bereitgestelltes Startformular wird mit Windows-typischen Komponenten ausgestattet.)

2. **Objekteigenschaften zuweisen**

   (Zuweisung der Eigenschaften, die sich während der Laufzeit verändern - z.B. Inhalt der Textfelder.)

3. **Ereignisbehandlung definieren**

   (Festlegung der Reaktion von Komponenten auf Ereignisse - z.B. Änderung der Beschriftung.)

➡ d.h.: hier beginnt die eigentliche Programmierarbeit!

Grundlegendes zum Programmieren mit VB

Prof. Dr. Aris Christidis

Auf der Grundlage eines Programms kann ein Rechner

- **Daten** mit seiner Umgebung **austauschen**,
- mathematische und logische **Operationen durchführen**,
- in Abhängigkeit von anfallenden Daten den **Programmablauf** zur Laufzeit **verändern**.

Programmiersprachen (VB) bieten Anweisungen (mind.)

- zur Daten-Eingabe, -Ausgabe und -Repräsentation,
- zu Operationen an Daten,
- zur gezielten Veränderung des Programmablaufs – aber auch

  zur Ablage verwendeter und zur Verwendung abgelegter Daten.

Grundlegendes zum Programmieren mit VB

Prof. Dr. Aris Christidis

Beispiel:

```
Sub cmdNetto_Click()
'Variable, Ausgabe:
Dim netto
Cls
Print "Netto von.."

'Operation:
netto = Brutto.Text / 1.16

'Bedingte Anweisung:
If netto < 0 Then Exit Sub

'Daten ablegen:
Open "MwSt.txt" For Output As #1
Print #1, "Netto:"; netto; _
          "von "; Brutto.Text
Close #1
End Sub
```

Programmiersprachen (VB) bieten Anweisungen (mind.)

- zur Daten-Eingabe, -Ausgabe und -Repräsentation,
- zu Operationen an Daten,
- zur gezielten Veränderung des Programmablaufs – aber auch

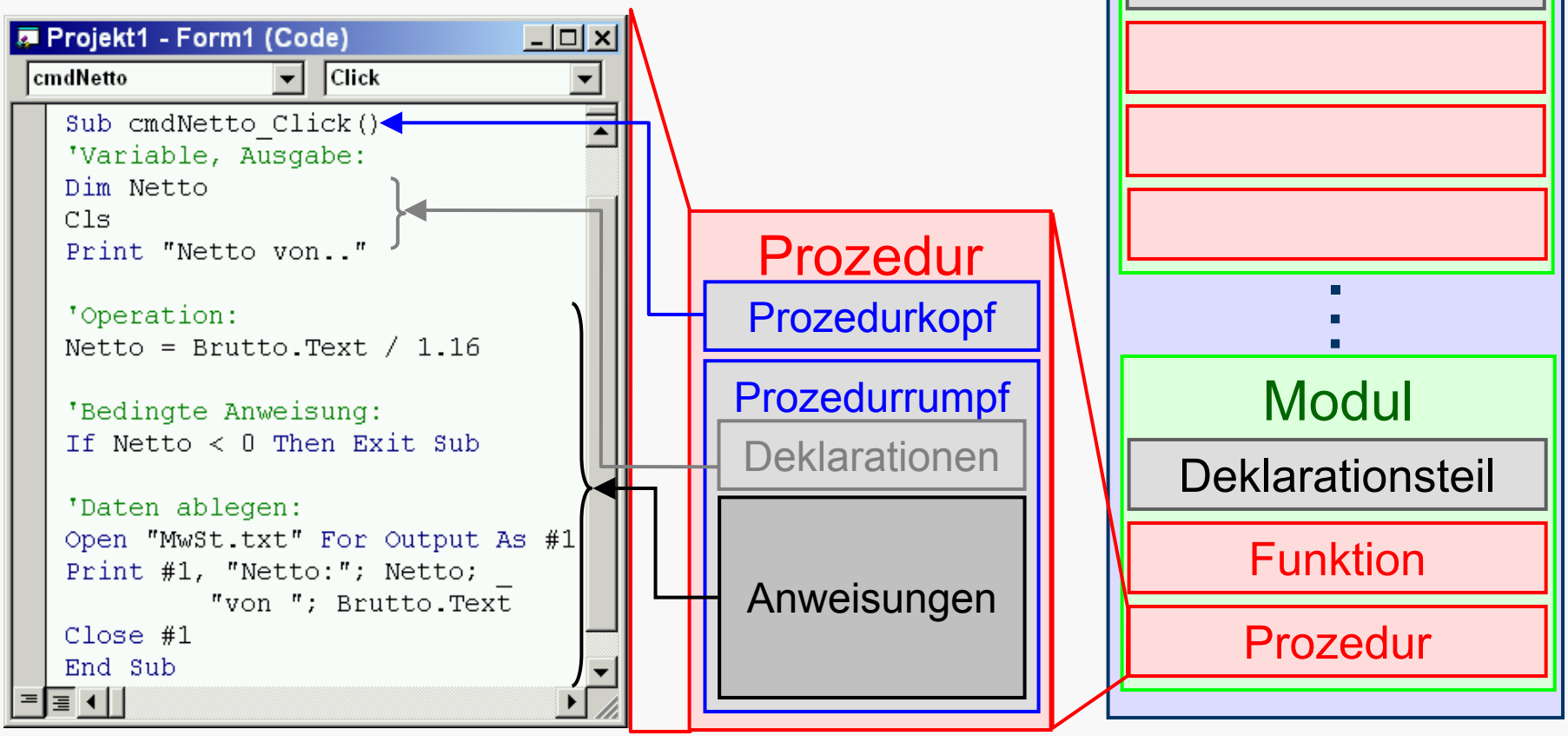
zur Ablage verwendeter und zur Verwendung abgelegter Daten.

(00Bsp.exe)

Grundlegendes zum Programmieren mit VB

Prof. Dr. Aris Christidis

- Ein **Programm** enthält mind. ein **Modul** (Code-Datei).
- **Module** enthalten mind. eine **Prozedur** bzw. eine **Funktion** (Code-Teil).



Grundlegendes zum Programmieren mit VB

Prof. Dr. Aris Christidis

Programme in (problemorientierten) Hochsprachen müssen in Maschinensprache übersetzt werden; diese Aufgabe erledigt ein Programm:

- Ein **Interpreter** übersetzt das (VB-)Programm **zeilenweise zur Laufzeit**.
  - **Vorteil:** „Rapid Prototyping“ möglich
  - **Nachteil:** Langsame Ausführung, z.T. Mehrfachübersetzung
- Ein **Compiler** übersetzt ein Programm **vollständig** in Maschinencode und **optimiert** es nach wählbaren Kriterien, **bevor** es laufen kann.
  - **Vorteil:** Beträchtlicher Geschwindigkeitsgewinn
  - **Nachteil:** Ausgereiftes Konzept wird vorausgesetzt

Grundlegendes zum Programmieren mit VB

Prof. Dr. Aris Christidis

VB-Zeichenvorrat:

Für selbstdefinierte Namen (z.B. Variablen-Namen) zulässig:

- Das lateinische Alphabet
- Die Ziffern 0...9
- Der Unterstrich "_"

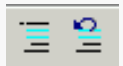
Unzulässig sind:

- Alle anderen Zeichen, da entweder Bestandteil von VB ($, %, #, ...), oder nicht definiert (ä, ß, ...)
- Alle Zeichenkombinationen, die nicht mit einem Buchstaben beginnen, länger als 256 Zeichen o. VB-Schlüsselwörter sind

Unberücksichtigt bleibt (obwohl zulässig):

- Unterscheidung Groß-/Kleinbuchstaben (LeseHilfe)
- alles, was rechts von **'** oder von **REM** steht (Kommentar)

⇨ Programmänderungen lieber durch Auskommentieren

(auch über: Ansicht / Symbolleisten / Bearbeiten / )

Grundlegendes zum Programmieren mit VB

Prof. Dr. Aris Christidis

Zur Notation von VB:

- Schlüsselwörter beginnen mit einem Großbuchstaben:
  - z.B.: **`Public`**, **`Private`**, ...
- Fortsetzungszeile mit Leerzeichen und Unterstrich “ _” ankündigen (nicht gültig für Zeichenketten!)
- Mehrere Anweisungen pro Zeile durch “**:**” trennen (nicht gültig für **`if`**-Anweisungen!)

Zur Notation der Erläuterungen:

- Ausdrücke groß schreiben:
  - z.B.: **`If`** `BEDINGUNG` **`Then`** `ANWEISUNGEN` **`End If`**
- Optionale Wörter in eckige Klammern setzen:
  - z.B.: `[`**`Option1`**`]`
- Alternative Wörter durch senkrechten Strich trennen:
  - z.B.: **`{ Alternative1`** | **`Alternative2 }`**

Sprachelemente von VB

Prof. Dr. Aris Christidis

- **Datentypen und Variablen**
- **Operatoren und Operationen**
- **Kontrollstrukturen:**
  - **Verzweigungen**
  - **Schleifenanweisungen**

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Variablen...

- ...sind Platzhalter für Werte, die während der Ausführung eines Programms benötigt (u. bei Bedarf verändert) werden
- ...werden beim Programm-Start auf 0 initialisiert (Strings sind leer)
- ...haben drei Eigenschaften:
  - Name (aus d. Zeichenvorrat, beginnend mit einem Buchstaben)
  - Datentyp
  - Gültigkeitsbereich

Beispiel:

```
Dim i As Integer, breite As Single, a As String
'äquivalent zu:
Dim i%, breite!, a$
```

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Datentypen:

- **`Boolean`**
  Wahrheitswert
- **`Byte`**, **`Integer`** (`%`), **`Long`** (`&`)
  Ganze Zahl
- **`Single`** (`!`), **`Double`** (`#`)
  Gleitkommazahl
- **`String`** (`$`)
  Zeichenkette
- **`Date`**
  Datum
- **`Currency`** (`@`)
  Rationale Zahl mit exakt vier Nachkommastellen
- **`Object`**
  Referenz auf Objekt
- **`Variant`** (mit Zeichenkette)
  beliebig

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Typ	kz	Erläuterung, Wertebereich	Speicher [Byte]
`Byte`		Ganzzahl, 0 ... 255	1
`Boolean`		Wahrheitswert, True = -1, False = 0	2
`Integer`	%	Ganzzahl, -32768 ... 32767	2
`Long`	&	Ganzzahl, lang: -2 147 483 648 ... 2 147 483 647	4
`Single`	!	Gleitkommazahl, einfachgenau (7 Ziffern) 10E-38 ... 10E+38	4
`Double`	#	Gleitkommazahl, doppeltgenau (16 Ziff.) 10E-308 ... 10E+308	8
`Currency`	@	Währung: (15 Stellen), (Dezimalpunkt), (4 Stellen)	8
`String`	$	Zeichenfolge, max. ca. 2 Mrd Zeichen (2^31)	10+Länge
`Date`		1. Januar 100 ... 31. Dezember 9999	8
`Object`		Verweis auf ein Objekt	4
`Variant`		universeller Datentyp	≥16

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Der Datentyp einer Variablen legt fest,

- welchen Wert (Wertebereich) die Variable haben kann
- welche Operationen mit der Variablen möglich sind.

  Beispiel: Variablen vom Typ **`Integer`** kann man addieren – nicht aber jene vom Typ **`String`**; diese kann man verketten („konkatenieren“).
- was für Werte der Variablen zugewiesen werden dürfen

  Beispiel: Ein **`Double`**-Wert kann einer **`Integer`**-Variablen zugewiesen werden; er wird aber an der 0,5-Schwelle auf- oder abgerundet.

  Umgekehrt kann ohne Verlust zugewiesen werden.

  Ein (Buchstaben-)**`String`** kann nicht z.B. einer **`Long`**-Variablen zugewiesen werden (Laufzeitfehler)

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

- **`Option Explicit`** als erste Zeile im Code erzwingt eine Überprüfung, ob verwendete Variablen auch deklariert wurden.

- Variablen sollten:
  - einen selbsterklärenden Namen haben
  - kommentiert werden
  - wenn möglich: Nie global sein
  - einigen Konventionen genügen, z.B.:
    - Schleifenvariablen heißen **`i`**, **`j`**, **`k`**, **`l`** (je nach Schleifentiefe)
    - Globale Variablen beginnen mit „**`g`**“ und dann mit Großbuchstabe
    - ... (auch eigenen) ...

Keine Unterscheidung von Groß- und Kleinschreibung

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Besonderheiten von Datentypen u. Variablen:

- Bei **`Single`** und **`Double`** ist der Dezimalpunkt einzugeben (aber ein Dezimalkomma wird ausgegeben).
- **`Single`**- oder **`Double`**-Werte, die **`Integer`**-Variablen zugewiesen werden, werden je nach Höhe des Nachkomma-Anteils ab- oder aufgerundet (z.B.: 0,4 zu 0 – aber 0,5 zu 1).
- **`Boolean`** codiert ‚falsch' als 0, ‚wahr' als -1:
  ```
  Dim var As Boolean : var = -1 '"wahr"
  ```
- Wertzuweisung für **`String`** in **“ “** einschließen:
  ```
  Dim s$ : s = “Das ist ein String!“
  ```
- Vereinbarung für **`String`** fester Länge als **`String*`**LÄNGE:
  ```
  Dim s As String * 3
  s = “Auch ein String!“ 'behält nur "Auc"
  ```
- Wertzuweisung für **`Date`** in **`#  #`** einschließen:
  ```
  Dim dat As Date : dat=#11/16/00#
  ```

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Besonderheiten von Datentypen u. Variablen: (Forts.)

- **`Variant`**-Variablen lassen sich als Zahlen oder als Zeichen interpretieren:

```
Dim tstVar As Variant
Dim tstInt As Integer
Dim tstStr As String
tstVar = 50
tstInt = tstVar + 50  '= 100 (Integer!)
tstStr = tstVar +"50" '="100"(String!)
```

- Mathematische Ausdrücke mit Variablen unterschiedlichen Typs werden im jeweils „mächtigsten“ Typ berechnet.
- Ausdrücke, die nur Zahlen (keine Variablen) enthalten, werden
  - als **`Double`** berechnet, wenn darin Zahlen mit Nachkomma-Stellen vorkommen,
  - als **`Integer`** berechnet, wenn darin nur ganze Zahlen vorkommen – auch, wenn das Ergebnis einem anderen Typ zugewiesen werden soll (Bsp. s.u.).

Sprachelemente von VB: Datentypen, Variablen

Prof. Dr. Aris Christidis

Gültigkeitsbereiche von Variablen:

- Deklaration für die gesamte **Applikation** muß auf Modul-Ebene mit **`Public`** erfolgen.
- Deklaration für ein **Modul** ist auch mit **`Private`** möglich.
- Deklaration mit **`Dim`** gilt im **Abschnitt**, in dem sie steht (Modul bzw. Funktion / Prozedur).

  Deklaration mit **`Static`** (statt **`Dim`**) in einer Funktion/ Prozedur bewirkt, daß beim Wiedereintritt der letzte Variablen-Wert zur Verfügung steht.

Deklarationen auf **Modulebene**

Deklarationen auf **Prozedurebene**